

# Random Forest for Label Ranking

Yangming Zhou<sup>a,b</sup> and Guoping Qiu<sup>a,c,\*</sup>

<sup>a</sup>*School of Computer Science, University of Nottingham Ningbo China, China*

<sup>b</sup>*LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France*

<sup>c</sup>*School of Computer Science, University of Nottingham, United Kingdom*

---

## Abstract

Label ranking aims to learn a mapping from instances to rankings over a finite number of predefined labels. Random forest is a powerful and one of the most successfully general-purpose machine learning algorithms of modern times. In the literature, there seems no research has yet been done in applying random forest to label ranking. In this paper, We present a powerful random forest label ranking method which uses random decision trees to retrieve nearest neighbors that are not only similar in the feature space but also in the ranking space. We have developed a novel two-step rank aggregation strategy to effectively aggregate neighboring rankings discovered by the random forest into a final predicted ranking. Compared with existing methods, the new random forest method has many advantages including its intrinsically scalable tree data structure, highly parallel-able computational architecture and much superior performances. We present extensive experimental results to demonstrate that our new method achieves the best predictive accuracy performances compared with state-of-the-art methods for datasets with complete ranking and datasets with only partial ranking information.

*keywords:* Label ranking, random forest, decision tree, nearest neighbour

---

## 1 Introduction

Label ranking aims to learn a mapping from instances to rankings over a finite set of predefined labels. It extends the conventional classification and multi-label classification in the sense that it needs to predict a ranking of all class labels instead of only one or several class labels. Both classification and

---

\* Corresponding author.

*Email addresses:* [yangming@info.univ-angers.fr](mailto:yangming@info.univ-angers.fr) (Yangming Zhou),  
[guoping.qiu@nottingham.ac.uk](mailto:guoping.qiu@nottingham.ac.uk) (Guoping Qiu).

multi-label classification can be considered as a special case of label ranking learning. Specifically, when only the top ranked label is required, label ranking reduces to a classification problem, when a calibrated label is introduced, label ranking is equivalent to a multi-label classification problem [1]. Due to its generality, label ranking has found in many practical applications such as natural language processing, recommender systems, bioinformatics and meta-learning [2].

The topic of label ranking has attracted increasing attention in the recent machine learning literature, and a large number of methods have been proposed or adapted for label ranking [3,4,5,6,7,8,9]. An overview of label ranking algorithms can be found in [10,11]. Existing methods can be mainly divided into two categories. One is known as reduction approaches which transform the label ranking problem into several simpler binary classification problems, and then the solutions of these classification problems are combined into a predicted ranking. Label ranking by learning pairwise preferences and by learning utility functions are two widely used schemes in the reduction approaches. For example, ranking by pairwise comparison (RPC) learns binary models for each pair of labels, and the predictions of these binary models are then aggregated into a ranking [4]; while constraint classification (CC) and log-linear models for label ranking (LL) seek to learn linear utility functions for each individual label instead of preference relations for each pair of labels [12,13]. Another category is probabilistic approaches which represent label ranking based on statistical models for ranking data, i.e., parametrized probability distributions on the class of all rankings. For example, Cheng et al. have developed instance-based (IB) learning algorithms based on the Mallows (M) and Plackett-Luce (PL) models [5,6], and Zhou et al. proposed a label ranking method based on Gaussian mixture models [7].

Both reduction approaches and probabilistic approaches have shown good performances in the experimental studies, while they also come with some disadvantages. For reduction approaches, theoretical assumptions on the sought “ranking-valued” mapping, which may serve as a proper learning bias, may not be easily translated into corresponding assumptions for the classification problems. Moreover, it is often not clear that minimizing the loss function on the binary problems leads to maximizing the performance of the label ranking model in terms of the desired loss function on rankings [14]. For probabilistic approaches, their success also do not come for free but at a large cost associated with both memory and time. For example, the instances-based approaches involve costly nearest neighbour search and the aggregation of neighboring rankings is also slow as it requires using complex optimization procedures, such as the approximate expectation maximization in IB-M and the minorization maximization in IB-PL [15]. Both IB-M and IB-PL are lazy learners, with almost no cost at training phase but a higher cost at predicting phase. It can be costly or even impossible in the resources-constrained

applications.

In this paper, to overcome these problems to some extent, we propose a label ranking method based on random forests (LR-RF). Random forests has been widely used to solve a number of machine learning, computer vision and medical image analysis tasks, and has achieved excellent performances [16,17]. The random forest has many advantages, which make it competitive for solving label ranking problem to existing approaches. Firstly, the tree structure of our LR-RF makes the retrieval of nearest neighbours more efficient than instance-based approaches. Secondly, rankings associated with the training instances are used as the supervising information to guide the construction of the random trees, thus enabling the retrieved nearest neighbours not only similar in feature space but also in the ranking space. Thirdly, both the construction and prediction processes of our method can be executed in a parallel way and is therefore able to achieve high computational efficiency on modern CPU or GPU hardware. To turn the random forest into an effective ranking method, we have proposed a simple and effective “top label as class” discretization method for discretizing ranking data, and we also developed a novel two-step rank aggregation strategy which effectively aggregate the neighboring rankings into a final predicted ranking. Experimental results show that our proposed LR-RF method achieves the best predictive accuracy performances compared with state-of-the-art methods for the case of training data with complete ranking and the case of data with partial ranking information.

The paper is organized as follows. Section 2 describes the label ranking problem in a more formal setting. Section 3 introduces several common distance measures for rankings. In Section 4, we present our label ranking method based on the random forest model. Section 5 is devoted to an experimental evaluation of the proposed LR-RF based on benchmark instances. Finally, Section 6 concludes the paper.

## 2 Label Ranking

Label ranking can be considered as a natural extension of the conventional classification problem. Given an instance  $x$  from an instance space  $\mathcal{X}$ , instead of predicting one or several possible class labels, label ranking tries to associate  $x$  with a total order of all class labels. This means that there exists a complete, transitive and asymmetric relation  $\succ_x$  on  $\mathcal{L}$ , where  $\lambda_i \succ_x \lambda_j$  shows that  $\lambda_i$  precedes  $\lambda_j$  in the ranking assigned to  $x$ .

We can identify a ranking  $\succ_x$  with a permutation  $\pi_x$  on  $\{1, 2, \dots, m\}$  such that  $\pi_x(i) = \pi_x(\lambda_i)$  is the position of  $\lambda_i$  in the ranking. This permutation

encodes the ranking given by

$$\lambda_{\pi_x^{-1}(1)} \succ_x \lambda_{\pi_x^{-1}(2)} \succ_x \dots \succ_x \lambda_{\pi_x^{-1}(m)} \quad (1)$$

where  $\pi_x^{-1}(i)$  is the index of the class label at position  $i$  in the ranking. For example, given a label set  $\mathcal{L} = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\}$ , and an observation over these labels  $\lambda_4 \succ \lambda_2 \succ \lambda_3 \succ \lambda_5 \succ \lambda_1$ , then we can represent this ranking by a permutation  $[5 \ 2 \ 3 \ 1 \ 4]$ . The set of all permutations of  $\{1, 2, \dots, m\}$  is denoted by  $\Omega$ . By abuse of terminology, we refer to elements  $\pi \in \Omega$  as both permutations and rankings.

Like in classification, we do not assume the existence of a deterministic  $\mathcal{X} \rightarrow \Omega$  mapping. Instead, every instance is associated with a probability distribution over  $\Omega$  [5]. It means that, for each  $x \in \mathcal{X}$ , there exists a probability distribution  $\mathcal{P}(\cdot|x)$  such that, for every  $\pi \in \Omega$ ,  $\mathcal{P}(\pi|x)$  is the probability that  $\pi$  is the ranking associated with  $x$ . The objective of label ranking is to learn a model in the form of a mapping  $\mathcal{X} \rightarrow \Omega$ . Generally, training data consists of a set of instances  $\mathcal{T} = \{\langle x_i, \pi_i \rangle\}$ ,  $i = 1, 2, \dots, n$ , where  $x_i$  is the feature vector containing the value of  $d$  feature attributes describing instance  $i$ , and  $\pi_i$  is the corresponding target ranking. Ideally, complete rankings are given as training information. However, it is much more important to allow for incomplete ranking information in the form of a partial ranking

$$\lambda_{\pi_x^{-1}(1)} \succ_x \lambda_{\pi_x^{-1}(2)} \succ_x \dots \succ_x \lambda_{\pi_x^{-1}(m')} \quad (2)$$

where  $m' < m$  and  $\{\pi_x^{-1}(1), \pi_x^{-1}(2), \dots, \pi_x^{-1}(m')\} \subset \{1, 2, \dots, m\}$ . In the above example, it is possible that only partial ranking information is provided for instance  $x$ , i.e.,  $\lambda_4 \succ_x \lambda_3 \succ_x \lambda_1$ , while no preference information is available for  $\lambda_2$  and  $\lambda_5$ .

### 3 Distance Measures

To evaluate the predictive performance of a label ranking algorithm, a suitable evaluation function is necessary. Kendall tau distance [18] is one of the most widely used distance measures for rankings. It essentially measures the total number of discordant label pairs (label pairs that are ranked in the opposite order in two rankings). Formally,

$$\mathcal{D}_K(\pi, \sigma) = \#\{(i, j) | \pi(i) > \pi(j) \wedge \sigma(i) < \sigma(j)\} \quad (3)$$

where  $1 \leq i < j \leq m$ . Kendall tau distance is an intuitive and easily interpretable performance measure. The time complexity of computing the Kendall tau distance between two rankings is  $\mathcal{O}(m \log m)$ . By normalizing Kendall tau

distance to the interval  $[-1, 1]$ , we can obtain Kendall's tau coefficient,

$$\tau = 1 - \frac{4\mathcal{D}_K(\pi, \sigma)}{m(m-1)} \quad (4)$$

which is a well-known correlation measure. Kendall's tau coefficient measures the proportion of the concordant pairs of labels in two rankings. Therefore, this measure can still work with partial rankings, as long as there is at least one pair of labels per instance. When  $\tau = 1$ , it means that the labels in ranking  $\pi$  and  $\sigma$  are sorted in the same order, while  $\tau = -1$  indicates that the labels in these two rankings are sorted in opposite order. In label ranking, performance comparisons among label ranking algorithms are often based on Kendall's tau coefficient. Two alternative distance measures on rankings include the Spearman distance

$$\mathcal{D}_S(\pi, \sigma) = \sum_{i=1}^m (\pi(i) - \sigma(i))^2 \quad (5)$$

and the Spearman footrule distance

$$\mathcal{D}_F(\pi, \sigma) = \sum_{i=1}^m |\pi(i) - \sigma(i)| \quad (6)$$

Both this two kinds of Spearman distances between two rankings can be computed in linear time  $\mathcal{O}(m)$ . Additionally, all the three distance measures can be extended in a natural way to several rankings. For example, the generalized Kendall distance between a complete  $\pi$  and a set of rankings  $\sigma_1, \dots, \sigma_k$  is given by

$$\mathcal{D}_K(\pi, \sigma_1, \dots, \sigma_k) = \sum_{i=1}^k \mathcal{D}_K(\pi, \sigma_i) \quad (7)$$

## 4 Random Forest for Label Ranking

Random forest is a powerful learning algorithm proposed in [19], which combines several randomized decision trees and aggregates their predictions by averaging. It has been one of the most successful general-purpose algorithms in modern times. In this section, we present a label ranking method based on random forest model, denoted as LR-RF. The proposed LR-RF works in two phases. It first constructs multiple decision trees by using different training instances at construction phase (Section 4.1), and then at the prediction phase, query instance passes through all trees, a two-step rank aggregation strategy is applied to aggregate the neighboring rankings into a final predicted ranking (Section 4.2). Fig. 1 illustrates the entire process from a query instance  $x$  to finally obtain the predicted ranking  $\hat{\pi}$

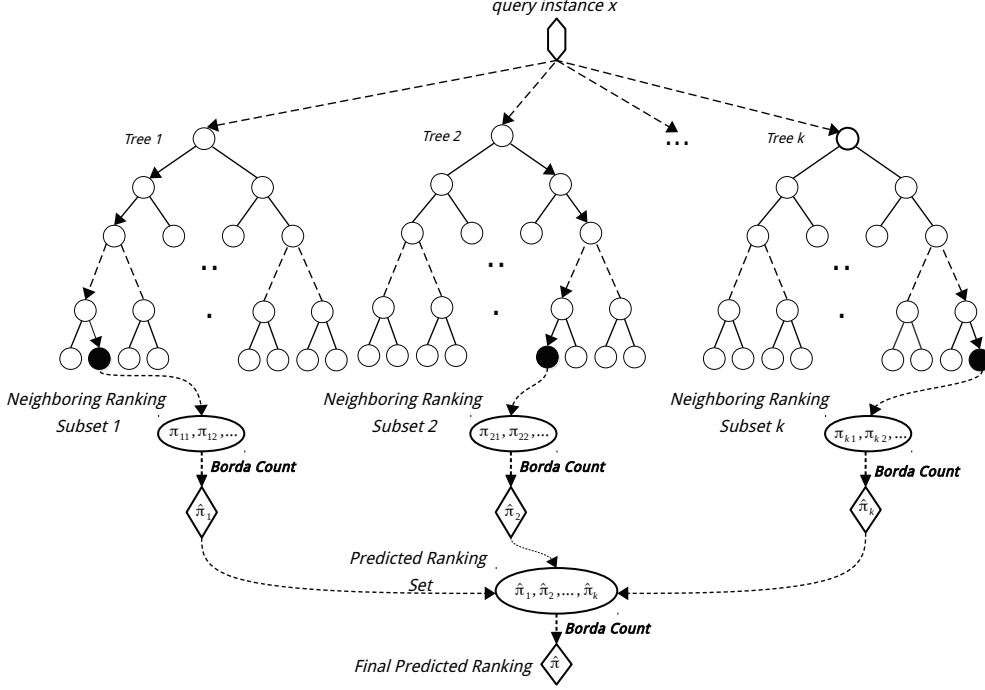


Fig. 1. Schematic illustration of random forest for label ranking.

#### 4.1 Construction of the Random Forest

To build a random forest, we need to generate  $nbr_{tree}$  different training data sets  $\mathcal{T}_i, i = 1 \dots nbr_{tree}$ , and each one is used to train a decision tree independently. Each training data set is drawn at random, with replacement, from the original data set  $\mathcal{T}$ . Then a decision tree is grown on this new training dataset. Specifically, at each node of each tree, a split is performed by maximizing the information gain  $\mathcal{G}$  over  $n_s = \lfloor \log_2 d \rfloor + 1$  attributes chosen uniformly at random among the  $d$  original attributes. Finally, construction of individual tree is stopped until one of the stopping conditions is satisfied. In the forest, all the trees grown are not pruned. To generate such a tree, we partition the training data set  $\mathcal{T}$  in a recursive manner, using one-dimensional splits defined by thresholds for an attribute value. The split function at each node is defined as follows

$$\begin{cases} x^i \geq x_{threshold}^i & \text{go to left child} \\ \text{otherwise,} & \text{go to right child} \end{cases} \quad (8)$$

where  $x^i$  and  $x_{threshold}^i$  are split attribute and split threshold of a best split point, respectively. Based on the above split function, we can split the training data of current node into the left child node and the right child node accordingly. To narrow down the search space for the split function, we only use a small group of attributes to split on rather than using all attributes each time. As different split attributes or split thresholds will bring about different

partitions on the training data, it is necessary to find a best split point for each node. In our algorithm, we use the rankings associated with each instance as supervising information to find the best split and guide the growing of the random trees.

In recent years, many discretization techniques have been proposed for label ranking. The simplest is Ranking As Class (RAC) [20], which simply treats rankings as classes:  $\forall \pi_i \in \Omega, \pi_i \rightarrow \lambda_i$ . This allows the use of many supervised discretization methods developed for classification in label ranking problems. However, if the RAC approach is applied, the number of classes can be extremely large, up to a maximum number of  $m!$ , where  $m$  is the total number of labels. Additionally, two complicated and specially designed discretization methods have also been proposed for label ranking, such as Minimum description length principle for ranking data (MDLP-R)[20] and Entropy-based discretization for ranking (EDiRa) [21]. Specifically, EDiRa takes into account the properties of rankings: how many distinct rankings are present in current node, and how similar they are to each other.

It is difficult to determine which supervised discretization technique is the best because many evaluation measures can be used to evaluate their performance [22]. In fact, all aforementioned discretization methods can be used for discretizing label ranking data. In our case, we make a compromise between the simple and the complicated supervised discretization methods, and propose to use the top label of a ranking to determine a class, i.e., Top Label As Class (TLAC). TLAC replaces the rankings by classes only consider the top label in the rankings regardless of other labels. For example, in Table 1, even though example 2 and 3 have distinct rankings [2 3 1] and [3 2 1] respectively, TLAC assigns the same class  $\lambda_3$  to them because their top labels are same.

Table 1  
Some selected examples from *iris* data set.

TID	$x^1$	$x^2$	$x^3$	$x^4$	$\pi$	$\lambda^{TLAC}$
1	-0.556	0.250	-0.864	-0.917	[1 2 3]	$\lambda_1$
2	0.167	0.000	0.186	0.167	[2 3 1]	$\lambda_3$
3	0.222	-0.167	0.424	0.583	[3 2 1]	$\lambda_3$
4	0.056	0.167	0.492	0.833	[3 1 2]	$\lambda_2$
5	-0.611	-1.000	-0.153	-0.250	[2 1 3]	$\lambda_2$
6	-0.111	-0.167	0.085	0.167	[2 3 1]	$\lambda_3$

With the help of TLAC, the well-known information gain is become available in label ranking. Information gain is a widely-used splitting criterion to find the best split points in decision trees [23]. It essentially measures the change

of class entropy before and after the partition caused by a split point. A split point at each node is represented by a split attribute and corresponding split threshold. For each split point, the entropy of the original data is compared with the weighted sum of the entropy of data in the left node and the right node. Let  $\mathcal{G}$  be the information gain, then at the split node  $j$ , the information gain obtained by a split point  $\theta$  in attribute  $A$  is defined as

$$\mathcal{G}(A, \theta; \mathcal{T}_j) = E(\mathcal{T}_j) - \sum_{i \in \{l, r\}} \frac{|\mathcal{T}_j^i|}{|\mathcal{T}_j|} E(\mathcal{T}_j^i) \quad (9)$$

where  $|\mathcal{T}_j|$  represents the number of instances contained in current node  $j$ , while  $|\mathcal{T}_j^l|$  and  $|\mathcal{T}_j^r|$  are the number of instances on its left child node and the number of instances on the right child node, respectively, with the split point  $\theta$  in attribute  $A$ . The entropy for data in node  $j$  is defined as

$$E(\mathcal{T}_j) = - \sum_{\pi \in \Omega_j} p(\pi) \log p(\pi) \quad (10)$$

where  $p(\pi)$  represents the proportion of instances with ranking  $\pi$  in the data set  $\mathcal{T}_j$  and  $\Omega_j$  is the set of all distinct rankings in  $\mathcal{T}_j$ .

A good split means it minimizes the overall entropy in its left child node and right child node. This can be achieved by finding the best split point at node  $j$  by

$$\theta^* = \arg \max_{\theta \in \mathcal{S}_j} \mathcal{G}(A, \theta; \mathcal{T}_j) \quad (11)$$

where  $\mathcal{S}_j$  is the set of all potential split points at node  $j$ . Based on this split criteria, the decision tree always chooses a split point which effectively splits the data at current node. This can be done recursively until the depth of the tree reaches a maximum allowable depth  $d_{max}$  or the entropy of data in current node is less than  $\epsilon_0$ .

#### 4.2 Prediction based on Neighboring Rankings

Once the whole random forest is constructed, it can be used to predict the potential ranking associated with a query instance. During prediction phase, we pass a query instance through all trees simultaneously (starting at the root node) until it reaches the leaf nodes. We call training examples stored in the leaf node as the neighbors of the query sample, and the rankings associated with neighbors as neighboring rankings. The predicted ranking is then obtained by aggregating these neighboring rankings with a two-step rank aggregation procedure.

The problem to aggregate the rankings of neighbors into a ranking is known as *rank aggregation* [24]. Rank aggregation has been studied extensively in



the context of social choice theory and meta-Search [25]. Rank aggregation can be obtained by optimizing different rank distance measures. For example, when Kendall tau distance is optimized, the achieved rank aggregation is called *Kemeny optimal aggregation*. It has been shown that the *Kemeny optimal aggregation* is the best compromise ranking. However, finding a *Kemeny optimal aggregation* is NP-hard even when  $k = 4$  [26]. In our algorithm, we resort to an efficient procedure called *Borda's method* to approximately solve it. The Borda's method was originally applied to aggregate label rankings by Klaus Brinker and Eyke Hüllermeier [27]. The principle of Borda's method is shown in Lemma 1.

**Lemma 1 (Borda's method)** *Given a collection of complete rankings  $\sigma_1, \dots, \sigma_k$ , for each label  $\lambda_i \in \mathcal{L}$  and ranking  $\sigma_j$ , Borda's method first assigns a score  $s_{ij} = \sigma_j(i)$ , and then the average Borda score  $s_i$  is defined as  $\frac{1}{k} \sum_{j=1}^k s_{ij}$ . The labels are then sorted in decreasing order of their average Borda score, and ties are broken at random.*

The rank aggregation obtained by Borda's method is an optimal aggregation with respect to the Spearman distance  $\mathcal{D}_S$  [28]. Moreover, it has been shown that Kendall tau distance  $\mathcal{D}_K$  can be approximated very well by Spearman distance  $\mathcal{D}_S$  [29], i.e.,  $\frac{1}{\sqrt{m}} \mathcal{D}_K(\pi, \sigma) \leq \mathcal{D}_S(\pi, \sigma) \leq 2\mathcal{D}_K(\pi, \sigma)$ , where  $m$  is the number of labels in the ranking. It means that there is a close relation between Kendall tau distance and Spearman distance. Therefore, rank aggregation obtained by Borda's method can be a good approximation of the Kemeny optimal aggregation without much sacrifice of predictive performance.

A primary advantage of Borda's method is that it is computationally very easy, and it can perform the aggregation of  $k$  complete rankings with  $m$  labels in linear time  $\mathcal{O}(k \cdot m)$ . However, it also shows its shortcomings in generalising to partial rankings. To solve this problem, Cheng et al. [5] proposed a generalized Borda's method (as shown in Lemma 2) to extend traditional Borda's method to partial rankings by apportioning all the excess score equally among all missing candidates.

**Lemma 2 (Generalized Borda's method)** *Given a set of partial rankings  $\sigma_1, \dots, \sigma_k$ , for each label  $\lambda_i$  and partial ranking of  $m' < m$  labels, if it is a missing label, then it receives  $s_{ij} = (m+1)/2$  votes; if it is an existing label with rank  $r \in \{1, \dots, m'\}$ , then its Borda score is  $s_{ij} = (m' + 1 - r)(m + 1)(m' + 1)$ . The average Borda score  $s_i$  is defined as  $\frac{1}{k} \sum_{j=1}^k s_{ij}$ . The labels are then sorted in decreasing order of their average Borda score.*

In the prediction phase of our LR-RF, we use a two-step rank aggregation strategy to aggregate the neighboring ranking. At the first step, we consider each decision tree individually, and each decision tree makes the same contribution to generate the final predicted ranking. Therefore, we need to aggregate

---

**Algorithm 1** Random forest label ranking

---

```
1: Input: query  $x$ , training data  $\mathcal{T}$ , number of trees  $nbr_{tree}$  and tree depth  $d_{max}$ 
2: Output: a predicted ranking  $\pi_x$  for query  $x$ 
   /* Construction phase */
3: for  $i = 1$  to  $nbr_{tree}$  do
4:   create a new data set  $\mathcal{T}_i$  of size  $n$  from  $\mathcal{T}$ 
5:   repeat
6:     select  $n_s$  attributes from the original  $d$  attributes
7:     find the split attribute  $A$  and its best split point  $\theta^*$ 
8:     split the data  $\mathcal{T}_i$  according to the  $\theta^*$  in attribute  $A$ .
9:   until a stopping condition is met
10: end for
   /* Prediction phase */
11: for  $i = 1$  to  $nbr_{tree}$  do
12:   pass the query  $x$  through  $i$ -th tree
13:   find the nearest neighbors of  $x$  in the tree
14:   aggregate all the neighboring rankings into a predicted ranking  $\pi'_i$  by
      Borda's method (first-step)
15: end for
16: aggregate  $nbr_{tree}$  rankings  $\pi'_1, \dots, \pi'_{nbr_{tree}}$  into a final predicted ranking  $\pi_x$ 
      by Borda's method (second-step)
```

---

all the rankings of neighbors, and a rank aggregation can be obtained at each tree. At the second step, we aggregate all the predicted ranking of  $nbr_{tree}$  trees into a final predicted ranking. In the two-step rank aggregation procedure, we need to perform  $nbr_{tree} + 1$  aggregations in total. It is noteworthy that both the construction and prediction of each decision tree can be executed in parallel, thus achieving high computational efficiency on modern parallel CPU or GPU hardware. A detailed description of our LR-RF algorithm can be found in Algorithm 1.

## 5 Computational results

In this section, we present an empirical evaluation of the proposed LR-RF with main state-of-the-art label ranking methods. Additionally, learning curves and the effect of number of trees  $nbr_{tree}$  on LR-RF are investigated respectively.

## 5.1 Data Sets

In our experiments, we evaluate the proposed LR-RF on the label ranking data sets from the KEBI Data Repository. These data sets are obtained by transforming multi-class and regression data sets from the UCI repository of machine learning databases and the Statlog collection into label ranking data sets in two different ways. (A) For classification data, a naive Bayes classifier is first trained on the complete data set. Afterwards, for each instance, all the class labels in the data set are ordered according to their predicted class label probabilities, breaking ties by ranking the class label with lower index first. (B) For regression data, some numerical attributes are removed from the set of predictors, and each one is treated as a label. To obtain a ranking, the attributes are standardized and then ordered by size. A summary of the data sets and their characteristics is provided in Table 2 <sup>1</sup>.

Table 2

The description of experimental data sets (the type indicates the way in which the data set has been generated).

Data sets	type	#instances	#features	#labels
authorship	A	841	70	4
bodyfat	B	452	7	7
calhousing	B	37152	4	4
cpu-small	B	14744	6	5
elevators	B	29871	9	9
fried	B	73376	9	5
glass	A	214	9	6
housing	B	906	6	6
iris	A	150	4	3
pendigits	A	10992	16	10
segment	A	2310	18	7
stock	B	1710	5	5
vehicle	A	846	18	4
vowel	A	528	10	11
wine	A	178	13	3
wisconsin	B	346	16	16

## 5.2 Experimental Settings

Results were obtained in terms of Kendall’s tau coefficient from five repetitions of a ten-fold cross-validation. To model incomplete observations, we modified

<sup>1</sup> All these data sets are publicly available at website: <http://www.uni-marburg.de/fb12/kebi/research/repository/labelrankingdata>

the training data according to the following rule. Given a ranking, we associate each label with a random probability. If the probability is less than a fixed missing probability  $p_0$  ( $0 \leq p_0 \leq 1$ ), we delete it from the ranking and keep it in the ranking otherwise. Hence,  $p_0 \times 100\%$  of the labels in the original data sets will be deleted on average.

Our LR-RF <sup>2</sup> algorithms is implemented by MATLAB 2014b running on a PC with an Intel Core i5 processor (2.6 GHz and 8 GB RAM). The detailed descriptions and settings of main parameters are provided in Table 3. It is possible to obtain better results by fine tuning these parameters.

Table 3

Parameter settings of the proposed LR-RF algorithm.

Parameters	Description	Value
$nbr_{tree}$	number of decision trees	50
$d_{max}$	maximum depth of the decision tree	8
$\epsilon_0$	minimum split entropy	0.001

### 5.3 Experimental Results

In recent years, a variety of label ranking algorithms have been proposed in the literature [10,11]. The representative algorithms mainly include constraint classification (CC), log-linear model (LL), ranking by pairwise comparisons (RPC), instance-based methods with the Mallows model (IB-M) and the Plackett-Luce model (IB-PL), and label ranking trees (LRT) [5]. However, some source codes or executed programs of these algorithms are not available. We are only able to compare the performance of the proposed LR-RF algorithm with the reference algorithms RPC, IB-PL, and LRT by means of the WEKA-LR <sup>3</sup>. WEKA-LR is a label ranking extension for WEKA. It implements three label ranking algorithms, including RPC, IB-PL, and LRT, only in case of complete ranking information. For incomplete ranking information, it is still not available. In the following, we focus on comparing the proposed LR-RF algorithm with these three algorithms. We run these algorithm with default parameters in our platform. Specially, we report the results of IB-PL algorithm with four kinds of different neighborhood size (i.e.,  $\{5, 10, 15, 20\}$ ).

To analyse these results, we use a two-step non-parametric statistical methodology to perform performance comparisons [30]. Firstly, we conduct a *Friedman test* which makes the null hypothesis that all algorithms are equivalent.

<sup>2</sup> The source code of the proposed LR-RF algorithm will be made available upon the publication of the paper.

<sup>3</sup> The package WEKA-LR is publicly available at: <https://www.uni-marburg.de/fb12/kebi/research/software>

If the null hypothesis is rejected, we then proceed with a post-hoc test named *two tailed sign test* in order to compare methods in a pairwise way. More specifically, the critical value for Friedman test at the significance level 0.05 is  $F_{0.05}(6, 90) = 2.21$ . Therefore, the Friedman test rejects the null hypothesis if the computed  $F_F$  value is less than the critical value 2.21. At the same significance level, the critical value for two-tailed sign test in our experiment is 12. It means that a method is significantly better than another when it performs better on at least 12 out of all 16 data sets.

Table 4

Performance comparisons between the proposed LR-RF algorithms with state-of-the-art algorithms in case of complete ranking.

data sets	RPC	IB-PL(5)	IB-PL(10)	IB-PL(15)	IB-PL(20)	LRT	LR-RF
authorship	0.908(2.0)	-0.203(4.0)	-0.271(5.0)	-0.276(6.0)	-0.344(7.0)	0.887(3.0)	<b>0.913</b> (1.0)
bodyfat	0.282(2.0)	<b>0.410</b> (1.0)	0.095(6.0)	0.170(4.0)	0.070(7.0)	0.110(5.0)	0.185(3.0)
calhousing	0.244(7.0)	0.341(5.0)	0.295(6.0)	0.401(2.0)	<b>0.471</b> (1.0)	0.357(4.0)	0.367(3.0)
cpu-small	0.449(4.0)	0.400(6.0)	<b>0.515</b> (1.5)	0.490(3.0)	0.367(7.0)	0.423(5.0)	<b>0.515</b> (1.5)
elevators	0.749(3.0)	0.721(5.5)	0.721(5.5)	0.721(5.5)	0.721(5.5)	<b>0.756</b> (1.5)	<b>0.756</b> (1.5)
fried	<b>0.999</b> (1.0)	0.760(4.0)	0.508(5.0)	0.421(6.0)	0.394(7.0)	0.890(3.0)	0.926(2.0)
glass	0.885(7.0)	<b>1.000</b> (2.5)	<b>1.000</b> (2.5)	<b>1.000</b> (2.5)	<b>1.000</b> (2.5)	0.889(5.0)	0.888(6.0)
housing	0.670(3.0)	0.266(4.0)	-0.048(7.0)	-0.018(5.0)	-0.032(6.0)	<b>0.803</b> (1.0)	0.792(2.0)
iris	0.889(5.0)	<b>0.978</b> (1.5)	<b>0.978</b> (1.5)	0.858(6.0)	0.804(7.0)	0.960(4.0)	0.966(3.0)
pendigits	0.932(4.0)	<b>0.975</b> (1.0)	0.840(5.0)	0.812(6.0)	0.787(7.0)	0.943(2.0)	0.939(3.0)
segment	0.934(7.0)	<b>1.000</b> (2.5)	<b>1.000</b> (2.5)	<b>1.000</b> (2.5)	<b>1.000</b> (2.5)	0.953(6.0)	0.961(5.0)
stock	0.779(6.0)	0.904(2.0)	0.851(4.0)	0.808(5.0)	0.737(7.0)	0.889(3.0)	<b>0.922</b> (1.0)
vehicle	0.850(6.0)	0.929(3.0)	<b>0.940</b> (1.0)	0.930(2.0)	0.857(5.0)	0.831(7.0)	0.860(4.0)
vowel	0.652(4.0)	0.841(2.0)	0.451(5.0)	0.279(6.0)	0.162(7.0)	0.794(3.0)	<b>0.967</b> (1.0)
wine	0.903(6.0)	0.940(5.0)	<b>1.000</b> (1.0)	0.989(3.0)	0.996(2.0)	0.884(7.0)	0.953(4.0)
wisconsin	<b>0.633</b> (1.0)	0.612(2.0)	0.063(5.0)	-0.028(7.0)	0.049(6.0)	0.351(4.0)	0.478(3.0)
average rank	4.25	3.19	3.97	4.47	5.41	3.97	2.75
wins	13	9	10.5	11	12	12.5	*

Table 4 represents the comparative results between the proposed LR-RF algorithm and the state-of-the-art algorithms on data sets with complete ranking information. In this table, we order the algorithms for each data set separately, the best performing algorithm obtaining the rank of 1, the second best rank 2, and so on. In case of ties, average ranks are assigned. For example, both LRT and LR-RF simultaneously achieved the best performance (i.e., 0.756) on data set *elevators*, thus they obtain the equal rank  $(1 + 2)/2 = 1.5$ . Finally, the average rank of each algorithm is obtained by averaging the ranks of all 16 data sets. We also give the number of data sets that LR-RF achieves better performance than the corresponding reference algorithms, as shown in the bottom lines of the table. The best performance on each data set is in bold. We clearly observe that there is no a single algorithm can achieve the best performance across all data sets. Specifically, the proposed LR-RF algorithm

gives the best performance on 5 out of 16 data sets, and obtaining the smallest average ranks 2.75 (4.25, 3.97, 3.19, 3.97, 4.47 and 5.41 are respectively obtained by the reference algorithm RPC, LRT and IB-PL with neighborhood size 5, 10, 15, and 20).

We resort to two-step statistical method to check the significant difference among these algorithms. According to the *Friedman test*, we calculate  $F_F = 2.93$ . With seven algorithms and 16 data sets,  $F_F$  is distributed according to the  $F$  distribution with  $7 - 1 = 3$  and  $(7 - 1) \times (16 - 1) = 90$  degrees of freedom. The critical value of  $F(6, 90)$  at the significance level 0.05 is 2.21 ( $2.93 > 2.21$ ), so we reject the null-hypothesis. We then proceed with a post-hoc *two tailed sign test* in order to compare methods in a pairwise way. As we can see from the last row of the Table 4, the proposed LR-RF algorithm respectively wins 13, 12.5, 9, 10.5, 11 and 12 data sets than the reference algorithm RPC, LRT and IB-PL with neighborhood size 5, 10, 15, and 20). At a significance level 0.05, the critical value of the two-tailed sign test is 12. Consequently, LR-RF significantly performs better than RPC, LRT and IB-PL with neighborhood size 20. Although there is no significant difference between LR-RF and IB-PL with neighborhood size 5, 10 and 15, LR-RF is better on 9, 10.5 and 11 out of 16 data sets, which are just slightly smaller than the critical value. The differences are likely to become significant when increasing the number of data sets. Since we fix a set of parameters for all data sets, it is also possible that the proposed LR-RF algorithm obtains better performance by fine tuning these parameters on each instances.

#### 5.4 Compared with reported results of the state-of-the-art algorithms

Given that source codes or executable programs of reference algorithms are not available in case of partial ranking information. In this section, we compare the performance of the proposed LR-RF algorithm with results reported in the literature. By comparing the results reported and the results obtained by running WEKA-LR, we observe there are only some slight differences for RPC, IB-PL and LRT. Moreover, it is well known that the executed platform is only closely related to computational time of an algorithm. Consequently, it is also credible to compare the results (in terms of accuracy) of the proposed algorithm with results reported of the reference algorithms. The reference algorithms mainly include constraint classification (CC), log-linear model (LL), instance-based methods with the Mallows model (IB-M), and also RPC, IB-PL and LRT. It is worth noting that the results of all reference algorithm are taken directly from the literature [5] and [15], and the neighborhood size  $\{5, 10, 15, 20\}$  of the IB-PL and IB-M were determined through cross validation on the training set.

Table 5-7 show performance comparisons between the proposed LR-RF algorithm and six state-of-the-art label ranking algorithms on three cases: complete ranking ( $p_0 = 0.0$ ), missing labels ranking with  $p_0 = 0.3$  and missing labels ranking with  $p_0 = 0.6$ . Pairwise comparisons between two methods are also provided in terms of win statistics in Table 8.

Table 5

Performance comparisons between LR-RF with state-of-the-art algorithms on the case  $p_0 = 0.0$ .

$p_0$	data sets	Reduction methods			Probabilistic methods		Tree-based methods	
		CC	LL	RPC	IB-M	IB-PL	LRT	LR-RF
0.0	authorship	0.920(3.0)	0.657(7.0)	0.910(5.0)	<b>0.936</b> (1.5)	<b>0.936</b> (1.5)	0.882(6.0)	0.913(4.0)
	bodyfat	0.281(2.0)	0.266(3.0)	<b>0.285</b> (1.0)	0.229(5.0)	0.230(4.0)	0.117(7.0)	0.185(6.0)
	calhousing	0.250(5.0)	0.223(7.0)	0.243(6.0)	0.344(2.0)	0.326(3.0)	0.324(4.0)	<b>0.367</b> (1.0)
	cup-small	0.475(4.0)	0.419(7.0)	0.450(5.0)	0.496(2.0)	0.495(3.0)	0.447(6.0)	<b>0.515</b> (1.0)
	elevators	<b>0.768</b> (1.0)	0.701(7.0)	0.749(4.0)	0.727(5.0)	0.721(6.0)	0.760(2.0)	0.756(3.0)
	fried	<b>0.999</b> (1.5)	0.989(3.0)	<b>0.999</b> (1.5)	0.900(5.0)	0.894(6.0)	0.890(7.0)	0.926(4.0)
	glass	0.846(4.0)	0.818(7.0)	0.882(3.0)	0.842(5.0)	0.841(6.0)	0.883(2.0)	<b>0.888</b> (1.0)
	housing	0.660(6.0)	0.626(7.0)	0.671(5.0)	0.736(3.0)	0.711(4.0)	<b>0.797</b> (1.0)	0.792(2.0)
	iris	0.836(6.0)	0.818(7.0)	0.889(5.0)	0.925(4.0)	0.960(2.0)	0.947(3.0)	<b>0.966</b> (1.0)
	pendigits	0.903(6.0)	0.814(7.0)	0.932(5.0)	<b>0.941</b> (1.0)	0.939(2.5)	0.935(4.0)	0.939(2.5)
	segment	0.914(5.0)	0.810(6.0)	0.934(4.0)	0.802(7.0)	0.950(2.0)	0.949(3.0)	<b>0.961</b> (1.0)
	stock	0.737(6.0)	0.696(7.0)	0.777(5.0)	<b>0.925</b> (1.0)	0.922(2.5)	0.895(4.0)	0.922(2.5)
	vehicle	0.855(3.5)	0.770(7.0)	0.854(5.0)	0.855(3.5)	0.859(2.0)	0.827(6.0)	<b>0.860</b> (1.0)
	vowel	0.623(6.0)	0.601(7.0)	0.647(5.0)	<b>0.882</b> (1.0)	0.851(3.0)	0.794(4.0)	0.867(2.0)
	wine	0.933(5.0)	0.942(4.0)	0.921(6.0)	0.944(3.0)	0.947(2.0)	0.882(7.0)	<b>0.953</b> (1.0)
	wisconsin	0.629(2.0)	0.542(3.0)	<b>0.633</b> (1.0)	0.501(4.0)	0.479(5.0)	0.343(7.0)	0.478(6.0)
	average rank	4.13	6.00	4.16	3.31	3.41	4.56	2.44

According to the average ranks of each methods, we can calculate  $F_F = 5.68, 4.94$  and  $6.46$  for cases with complete ranking, 30% missing labels and 60% missing labels, respectively. All the three  $F_F$  values are larger than the critical value of the *Friedman test* 2.21. Consequently, the *Friedman test* rejects the null hypothesis in all three cases, which suggests significant differences among these seven methods in all three cases.

After rejecting the null-hypothesis, we conduct a *two-tailed sign test*. As we can see from Table 8, in case of complete ranking information, our method is significantly better than LL, RPC, IB-PL and LRT, winning 12, 13, 12 and 14 of the 16 data sets respectively. We also observe that our method LR-RF is better than CC and IB-M on 11 and 10 data sets, and they are just below the critical value 12. In case of partial ranking information, our method LR-RF is significantly better than six other methods. Specifically, when  $p_0 = 0.30$ , our LR-RF obtains better results than CC, LL, IB-M, IB-PL and LRT on 12, 13, 13, 14, 14 and 15 out of 16 data sets respectively; when  $p_0 = 0.60$ , our



Table 6

Performance comparisons between LR-RF with state-of-the-art algorithms on the case  $p_0 = 0.30$ .

$p_0$	data sets	Reduction methods			Probabilistic methods		Tree-based methods	
		CC	LL	RPC	IB-M	IB-PL	LRT	LR-RF
0.3	authorship	0.891(4.0)	0.656(7.0)	0.884(5.0)	0.913(2.0)	<b>0.927</b> (1.0)	0.871(6.0)	0.911(3.0)
	bodyfat	0.260(2.0)	0.251(3.0)	<b>0.272</b> (1.0)	0.198(5.0)	0.204(4.0)	0.097(7.0)	0.181(6.0)
	calhousing	0.249(5.0)	0.223(7.0)	0.243(6.0)	0.310(2.0)	0.303(4.0)	0.307(3.0)	<b>0.364</b> (1.0)
	cup-small	0.474(3.0)	0.419(6.0)	0.449(5.0)	0.473(4.0)	0.477(2.0)	0.405(7.0)	<b>0.513</b> (1.0)
	elevators	<b>0.767</b> (1.0)	0.699(6.0)	0.748(4.0)	0.683(7.0)	0.702(5.0)	0.756(2.0)	0.755(3.0)
	fried	0.998(2.0)	0.989(3.0)	<b>0.999</b> (1.0)	0.850(7.0)	0.861(6.0)	0.863(5.0)	0.924(4.0)
	glass	0.835(4.0)	0.817(5.0)	0.851(2.0)	0.776(7.0)	0.809(6.0)	0.850(3.0)	<b>0.886</b> (1.0)
	housing	0.655(5.0)	0.625(7.0)	0.667(4.0)	0.669(3.0)	0.654(6.0)	0.734(2.0)	<b>0.789</b> (1.0)
	iris	0.807(6.0)	0.804(7.0)	0.871(4.0)	0.867(5.0)	0.926(2.0)	0.909(3.0)	<b>0.962</b> (1.0)
	pendigits	0.902(5.5)	0.802(7.0)	0.932(2.0)	0.902(5.5)	0.918(3.0)	0.914(4.0)	<b>0.939</b> (1.0)
	segment	0.911(4.0)	0.806(6.0)	0.933(2.5)	0.735(7.0)	0.874(5.0)	0.933(2.5)	<b>0.961</b> (1.0)
	stock	0.735(6.0)	0.691(7.0)	0.776(5.0)	0.855(4.0)	0.877(2.5)	0.877(2.5)	<b>0.922</b> (1.0)
	vehicle	0.839(2.0)	0.769(7.0)	0.834(4.0)	0.822(5.0)	0.838(3.0)	0.819(6.0)	<b>0.859</b> (1.0)
	vowel	0.615(6.0)	0.598(7.0)	0.644(5.0)	0.810(2.0)	0.785(3.0)	0.718(4.0)	<b>0.851</b> (1.0)
	wine	0.911(5.0)	0.941(2.0)	0.902(6.0)	0.930(3.0)	0.926(4.0)	0.862(7.0)	<b>0.952</b> (1.0)
	wisconsin	<b>0.617</b> (1.0)	0.533(3.0)	0.607(2.0)	0.464(5.0)	0.453(6.0)	0.284(7.0)	0.468(4.0)
	average rank	3.84	5.63	3.66	4.51	3.91	4.44	1.94

LR-RF shows more powerful performance, and even dominates LRT on all 16 data sets. The following two conclusions can also be drawn from Table 3-5 and Table 8:

- Comparing two tree-based methods, our LR-RF is significantly better than LRT, winning 14, 15 and even 16 out of 16 data sets in the three cases, respectively. This results confirm the conclusions that a ensemble model (e.g., LR-RF) can produce substantial improvements in the performance compared with the use of a single model (e.g., LRT).
- Comparing performances of LR-RF in these three cases, we find that our LR-RF has stronger ability to deal with data set with partial rankings. When the probability of missing labels becomes higher, the advantage of the LR-RF is more obvious. When  $p_0 = 0$  increase to  $p_0 = 0.3$  and  $p_0 = 0.6$ , we find the average ranks of LR-RF decrease and achieves the best performance on much more data sets, and even obtains best results on all 16 data sets in the case of  $p_0 = 0.6$ .



Table 7

Performance comparisons between LR-RF with state-of-the-art algorithms on the case  $p_0 = 0.60$ .

$p_0$	data sets	Reduction methods			Probabilistic methods		Tree-based methods	
		CC	LL	RPC	IB-M	IB-PL	LRT	LR-RF
0.6	authorship	0.835(5.0)	0.650(7.0)	0.872(3.0)	0.849(4.0)	0.886(2.0)	0.828(6.0)	<b>0.897</b> (1.0)
	bodyfat	0.224(3.0)	<b>0.241</b> (1.0)	0.235(2.0)	0.160(4.0)	0.151(5.0)	0.007(7.0)	0.095(6.0)
	calhousing	0.247(5.0)	0.221(7.0)	0.242(6.0)	0.263(3.0)	0.259(4.0)	0.273(2.0)	<b>0.363</b> (1.0)
	cup-small	0.470(2.0)	0.418(6.0)	0.448(3.0)	0.429(5.0)	0.437(4.0)	0.367(7.0)	<b>0.511</b> (1.0)
	elevators	<b>0.765</b> (1.0)	0.696(5.0)	0.748(3.0)	0.596(7.0)	0.633(6.0)	0.742(4.0)	0.755(2.0)
	fried	<b>0.997</b> (1.5)	0.987(3.0)	<b>0.997</b> (1.5)	0.777(7.0)	0.797(6.0)	0.809(5.0)	0.921(4.0)
	glass	0.789(5.0)	0.808(2.0)	0.799(3.5)	0.611(7.0)	0.675(6.0)	0.799(3.5)	<b>0.867</b> (1.0)
	housing	0.638(3.0)	0.614(5.0)	0.641(2.0)	0.543(6.0)	0.492(7.0)	0.634(4.0)	<b>0.776</b> (1.0)
	iris	0.743(7.0)	0.768(6.0)	0.779(5.0)	0.799(3.0)	0.868(2.0)	0.794(4.0)	<b>0.959</b> (1.0)
	pendigits	0.900(3.0)	0.787(6.0)	0.929(2.0)	0.781(7.0)	0.794(5.0)	0.871(4.0)	<b>0.938</b> (1.0)
	segment	0.902(4.0)	0.801(5.0)	0.920(2.0)	0.612(7.0)	0.674(6.0)	0.903(3.0)	<b>0.957</b> (1.0)
	stock	0.724(5.5)	0.689(7.0)	0.771(3.0)	0.724(5.5)	0.740(4.0)	0.827(2.0)	<b>0.912</b> (1.0)
	vehicle	0.810(2.0)	0.764(5.5)	0.786(3.0)	0.736(7.0)	0.765(4.0)	0.764(5.5)	<b>0.854</b> (1.0)
	vowel	0.598(5.0)	0.591(6.0)	0.612(4.0)	0.638(2.0)	0.588(7.0)	0.615(3.0)	<b>0.787</b> (1.0)
	wine	0.853(6.0)	0.894(3.0)	0.864(5.0)	0.893(4.0)	0.907(2.0)	0.752(7.0)	<b>0.926</b> (1.0)
	wisconsin	<b>0.566</b> (1.0)	0.518(3.0)	0.536(2.0)	0.399(4.0)	0.381(5.0)	0.251(7.0)	0.371(6.0)
	average rank	3.69	4.84	3.13	5.16	4.69	4.63	1.88

Table 8

Pairwise comparisons of the methods in terms of win/win/win statistic: Wins in the complete rankings, in the 30% missing labels and in the 60% missing labels scenario.

Methods	CC	LL	RPC	IB-M	IB-PL	LRT	LR-RF
CC	*	15/15/12	6.5/7/5.5	6.5/8.5/10.5	5/8/11	8/8/10	5/4/4
LL	1/1/4	*	1/1/3	4/7/10	3/5/8	4/5/6.5	4/3/3
RPC	9.5/9/10.5	15/15/13	*	6/9/12	5/8/12	7/9/11	3/3/3
IB-M	9.5/7.5/5.5	12/9/6	10/7/4	*	10.5/5/5	11/8/7	6/2/2
IB-PL	11/8/5	13/11/8	11/8/4	5.5/11/11	*	13/9.5/7	4/2/2
LRT	8/8/6	12/11/9.5	9/7/5	5/8/9	3/6.5/9	*	2/1/0
LR-RF	11/12/12	12/13/13	13/13/13	10/14/14	12/14/14	14/15/16	*

### 5.5 Parameter analysis

The parameter analysis includes three sets of experiments. We study the learning curves of the proposed LR-RF algorithm, and conduct sensitive analysis on two important parameters, i.e.,  $nbr_{tree}$  and  $d_{max}$ . The following experimental analysis is based on a selection of 4 data sets (*glass*, *housing*, *iris*, and *wine*).

### 5.5.1 Learning curves

In this set of experiments, we perform experiments to gain some understanding about how the missing probability  $p_0$  affects the performance of LR-RF. We did experiments with LR-RF on four selected data sets for different missing probability  $p_0$ , varying  $p_0$  from 0.1 to 0.8 by steps of 0.1. Figure 2 shows the learning curves, which present the performance of LR-RF as a function of the missing probability  $p_0$ .

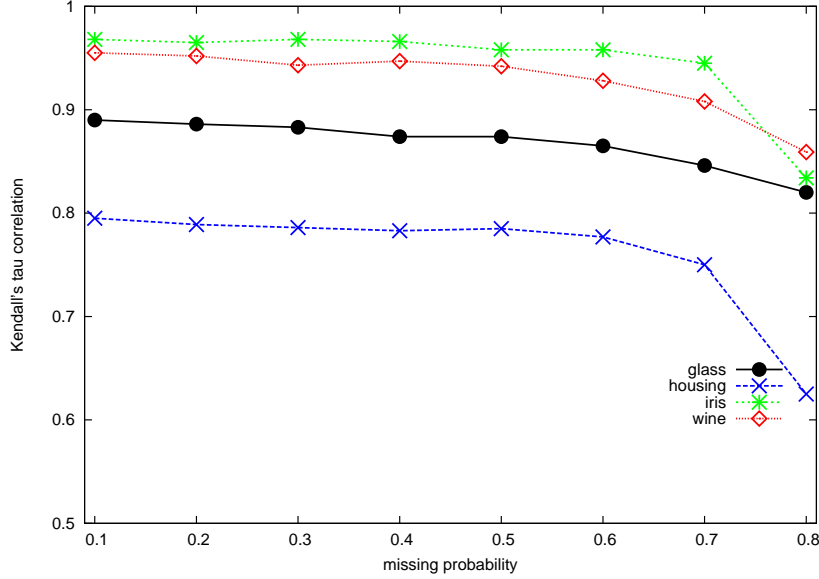


Fig. 2. Ranking performance (in terms of Kendall’s tau correlation) of the LR-RF as a function of the probability of missing label  $p_0$ .

It can be observed that the learning curves are relatively flat when  $p_0$  varies from 0.10 to 0.60. When the probability of missing label continues to increase, the performance of LR-RF rapidly worsening. The results indicate that our proposed method LR-RF has a strong ability to deal with data with missing label ranking information.

### 5.5.2 Sensitivity to the $nbr_{tree}$ parameter

The second set of experiments investigate how the parameter  $nbr_{tree}$  affects the accuracy of the proposed learning method LR-RF. We did experiments on data with complete ranking for different  $nbr_{tree}$  values, varying  $nbr_{tree}$  from 1 to 75 by steps of 5, and fixing other parameter value at the same time. Specifically, when  $nbr_{tree}$  equals to 1, the LR-RF is a single decision tree. It is same to the LRT but with different supervised discretization method to discrete ranking data.

In Figure 3, we are able to see how the different number of trees  $nbr_{tree}$  affects

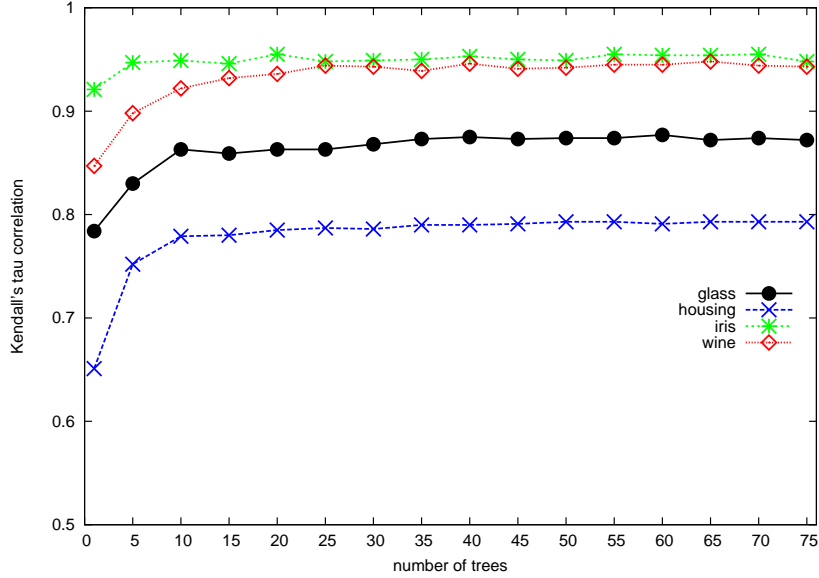


Fig. 3. Ranking performance (in terms of Kendall’s tau correlation) of the LR-RF as a function of the number of trees  $nbr_{tree}$ .

the performance of the LR-RF. It is interesting to find that, when  $nbr_{tree}$  increases from 1 to 10, the accuracy of the LR-RF significantly improves, while the accuracy of the LR-RF keeps relatively steady if  $nbr_{tree}$  continues to increase until 75. This analysis shows that a reasonable  $nbr_{tree}$  value plays an important role in the success of the LR-RF. According to the analysis, it is easy to determine a reasonable value for  $nbr_{tree}$ . In our proposed method LR-RF, we fix the total number of decision tree  $nbr_{tree}$  as 50.

### 5.5.3 Sensitivity to the $d_{max}$

The third set of experiments study how the tree depth  $d_{max}$  affects the performance of the LR-RF. We did experiments on data with complete ranking for different  $d_{max}$  values, varying  $d_{max}$  from 1 to 15 by steps of 1, and fixing other parameter value at the same time.

Figure 4 describes the influence of tree depth  $d_{max}$  on the performance of LR-RF. When the  $d_{max}$  is small, we observe that as the  $d_{max}$  increases the overall performance of LR-RF also increases, and then the performance keeps a relative steady value if we continue to increase the value of  $d_{max}$ . Moreover, the tree depth is a function of the problem complexity. For a given  $d_{max}$  value, the number of possible nodes is up to  $2^{d_{max}}$  in a random tree. Therefore, in our algorithm, we set  $d_{max}$  as 8.

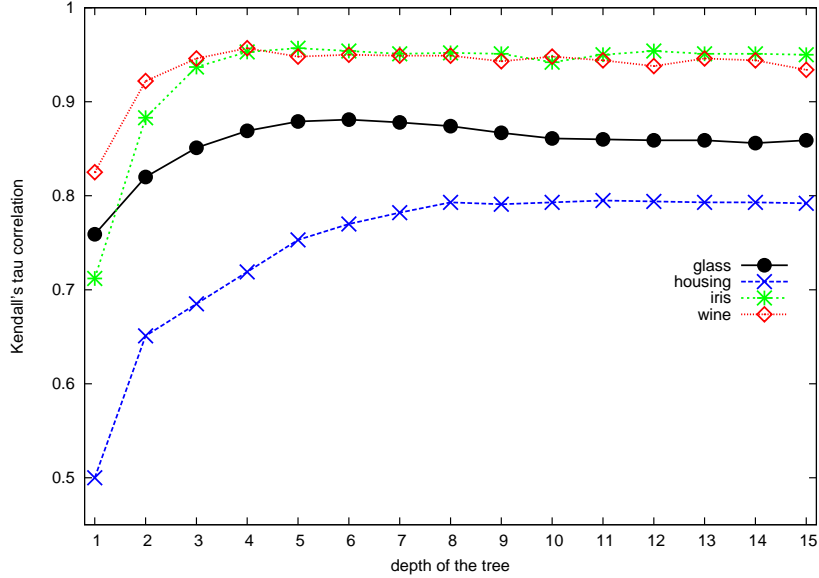


Fig. 4. Ranking performance (in terms of Kendall’s tau correlation) of the LR-RF as a function of the depth of tree  $d_{max}$ .

## 6 Conclusions and Future Work

In this paper, we have developed a novel label ranking method based on random forest (LR-RF). The empirical results show that our method is highly competitive with state-of-the-art methods in terms of predictive accuracy for datasets with complete ranking and datasets with partial ranking information. In addition to achieving state of the art performances, the new method has some further advantages. Firstly, the tree structure of our LR-RF makes the retrieval of nearest neighbours more efficient and scalable than traditional instance-based approaches. Secondly, rankings associated with the training instances are used as the supervising information to guide the construction of the random trees, thus enabling the retrieved nearest neighbours not only similar in feature space but also in the ranking space. Thirdly, both the construction and prediction processes of our method can be executed in a parallel way and is therefore able to achieve high computational efficiency on modern CPU or GPU hardware. Additionally, our LR-RF method is a tree-based method, tree structure can clearly express much more information about the problem and it is easy to understand even for people without a background on learning algorithms.

We also performed three sets of experiments to investigate the learning curves of the LR-RF in terms of missing probability, and the influence of parameter  $nbr_{tree}$  and  $d_{max}$  on the accuracy of the LR-RF, respectively. The analysis results show that our method has a strong robustness on the missing label information and parameter values selection.

For future work, we plan to compare and analyse the different supervised discretization methods, and select the suitable one to guide the growing of the decision trees in a random forest. It is possible to further improve the performance of LR-RF by use the suitable supervised discretization methods for discretizing label ranking data.

## References

- [1] J. Fürnkranz, E. Hüllermeier, E. L. Mencía, K. Brinker, Multilabel classification via calibrated label ranking, *Machine Learning* 73 (2) (2008) 133–153.
- [2] J. Fürnkranz, E. Hüllermeier, *Preference learning*, Springer, 2010.
- [3] S. Shalev-Shwartz, Y. Singer, Efficient learning of label ranking by soft projections onto polyhedra, *The Journal of Machine Learning Research* 7 (2006) 1567–1599.
- [4] E. Hüllermeier, J. Fürnkranz, W. Cheng, K. Brinker, Label ranking by learning pairwise preferences, *Artificial Intelligence* 172 (16) (2008) 1897–1916.
- [5] W. Cheng, J. C. Huhn, E. Hüllermeier, Decision tree and instance-based learning for label ranking, in: *Proc. ICML, 2009*, pp. 161–168.
- [6] W. Cheng, K. Dembczynski, E. Hüllermeier, Label ranking methods based on the plackett-luce model, in: *Proc. ICML, 2010*, pp. 215–222.
- [7] Y. Zhou, Y. Liu, X. Z. Gao, G. Qiu, A label ranking method based on gaussian mixture model, *Knowledge-Based Systems* 72 (2014) 108–113.
- [8] D. Schäfer, E. Hüllermeier, Dyad ranking using A bilinear plackett-luce model, in: *Proc. ECML/PKDD, 2015*, pp. 227–242.
- [9] E. Hüllermeier, W. Cheng, Superset learning based on generalized loss minimization, in: *Proc. ECML/PKDD, 2015*, pp. 260–275.
- [10] S. Vembu, T. Gärtner, Label ranking algorithms: A survey, in: *Preference learning*, Springer, 2011, pp. 45–64.
- [11] Y. Zhou, Y. Liu, J. Yang, X. He, L. Liu, A taxonomy of label ranking algorithms, *Journal of Computers* 9 (3) (2014) 557–565.
- [12] S. Har-Peled, D. Roth, D. Zimak, Constraint classification for multiclass classification and ranking, *Proc. NIPS* (2003) 809–816.
- [13] O. Dekel, C. D. Manning, Y. Singer, Log-linear models for label ranking, in: *Proc. NIPS, 2003*.
- [14] J. C. Duchi, L. W. Mackey, M. I. Jordan, On the consistency of ranking algorithms, in: *Proc. ICML, 2010*, pp. 327–334.

- [15] W. Cheng, Label ranking with probabilistic models, Ph.D. thesis, Philipps University Marburg (2012).
- [16] A. Criminisi, J. Shotton, E. Konukoglu, Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning, *Foundations and Trends in Computer Graphics and Vision* 7 (2-3) (2012) 81–227.
- [17] H. Fu, G. Qiu, Fast semantic image retrieval based on random forest, in: *Proc. ACM Multimedia*, 2012, pp. 909–912.
- [18] M. Kendall, *Rank Correlations Methods*, Griffin, 1955.
- [19] L. Breiman, Random forests, *Machine Learning* 45 (1) (2001) 5–32.
- [20] C. R. de Sá, C. Soares, A. Knobbe, P. Azevedo, A. M. Jorge, Multi-interval discretization of continuous attributes for label ranking, in: *Discovery Science*, Springer, 2013, pp. 155–169.
- [21] C. R. de Sá, C. Soares, A. Knobbe, Entropy-based discretization methods for ranking data, *Information Sciences* 329 (2016) 921–936.
- [22] S. Garcia, J. Luengo, J. A. Sáez, V. López, F. Herrera, A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning, *Knowledge and Data Engineering, IEEE Transactions on* 25 (4) (2013) 734–750.
- [23] J. R. Quinlan, Induction of decision trees, *Machine learning* 1 (1) (1986) 81–106.
- [24] S. Lin, Rank aggregation methods, *Wiley Interdisciplinary Reviews: Computational Statistics* 2 (5) (2010) 555–570.
- [25] W. W. Cohen, R. E. Schapire, Y. Singer, Learning to order things, *Journal of Artificial Intelligence Research* 10 (1) (1999) 243–270.
- [26] J. Bartholdi III, C. A. Tovey, M. A. Trick, Voting schemes for which it can be difficult to tell who won the election, *Social Choice and welfare* 6 (2) (1989) 157–165.
- [27] K. Brinker, E. Hüllermeier, Case-based label ranking, in: *European Conference on Machine Learning*, Springer, 2006, pp. 566–573.
- [28] E. Hüllermeier, J. Fürnkranz, Comparison of ranking procedures in pairwise preference learning, in: *Proc. IPMU*, 2004.
- [29] P. Diaconis, R. L. Graham, Spearman’s footrule as a measure of disarray, *Journal of the Royal Statistical Society. Series B (Methodological)* (1977) 262–268.
- [30] J. Demsar, Statistical comparisons of classifiers over multiple data sets, *Journal of Machine Learning Research* 7 (2006) 1–30.